- **Resp Time**: Average response time for the elapsed second
- **Err** (error rate): Errors occur when the server returns a response that is not the expected `HTTP 200 OK`
- **Count**: Total transaction count

You can fiddle with the number of simultaneous connections, and see how your server performs in order to establish a balanced configuration for your setup. Three tests were run here with different numbers of connections. The results speak for themselves:

	Test 1	**Test 2**	**Test 3**
Simultaneous connections	1	20	1000
Transactions per second (Tps)	67.54	205.87	185.07
Average response time	14 ms	91 ms	596 ms

Too few connections result in a low Tps rate; however, the response times are optimal. Too many connections produce a relatively high Tps, but the response times are critically high. You thus need to find a happy medium.

Upgrading Nginx gracefully

There are many situations where you need to replace the Nginx binary, for example, when you compile a new version and wish to put it in production, or simply after having enabled new modules and rebuilt the application. What most administrators would do in this situation is stop the server, copy the new binary over the old one, and start Nginx again. While this is not considered to be a problem for most websites, there may be some cases where the uptime is critical, and connection losses should be avoided at all costs. Fortunately, Nginx embeds a mechanism allowing you to switch binaries with uninterrupted uptime—zero percent request loss is guaranteed if you follow these steps carefully:

1. Replace the old Nginx binary (by default, `/usr/local/nginx/sbin/nginx`) with the new one.
2. Find the `pid` of the Nginx master process, for example, with `ps x | grep nginx | grep master`, or by looking at the value found in the pid file.
3. Send a `USR2` (12) signal to the master process—`kill -USR2 1234`, replacing `1234` with the pid found in *step 2*. This will initiate the upgrade by renaming the old `.pid` file and running the new binary.

4. Send a `WINCH` (28) signal to the old master process—`kill -WINCH 1234`, replacing `1234` with the `pid` found in *step 2*. This will engage a graceful shutdown of the old worker processes.
5. Make sure that all the old worker processes are terminated, and then send a `QUIT` signal to the old master process—`kill -QUIT 1234`, replacing `1234` with the `pid` found in *step 2*.

Congratulations! You have successfully upgraded Nginx and have not lost a single connection.

Summary

This chapter provided an overview of the configuration architecture by studying the syntax and the core module directives that have an impact on the overall server performance. We then went through a series of adjustments in order to fit your profile, followed by performance tests that have probably led you to fine-tune some more.

This is just the beginning though. Practically everything that we will be doing from now on is to prepare configuration sections. The next chapter will detail more advanced directives by further exploring the module system and the exciting possibilities that are offered to you.